Project Report

1. **Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

- The main direction of our project, which is to allow a user to find college recommendations ranked based on selected criteria, has stayed the same. However, there are differences in the final project compared to the original proposal.
- Our final product is similar to the original proposal as it contains the main features, which includes filtering based on metrics like Crime Rate. Due to time constraints, we did not include some categories like tuition costs, SAT, and GPA for the user to select from.
- Our final frontend design is fairly similar. The user can search for colleges by name using "Get School By Name". Although, our original design had a separate search bar at the top of the entire application. We also display a couple choices for the user to select from (CrimeRate, RateMyProf, HighRateMyProf + LowCrimeRate). This is displayed as buttons to query with, rather than the checkboxes shown in the original design. There is a submit button to display a table of colleges ranked accordingly, instead of a "Go lucky!" button as shown in our original design.
- We didn't implement the "login" functionality, as it is fairly difficult to do so.
- We didn't implement the checkboxes, because it is extremely difficult to implement without an ORM, so we implemented some pre-set fetches.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

- ACHIEVED: The user can easily find top U.S. colleges and universities based on simple criteria (such as Crime Rate, RateMyProf ranking)
- FAILED: In our opinion, we still need to add more features and buttons to make it more useful. For example, we can add more filter options (by MidCareerPay, STEMPercent, etc.) in order to further tailor towards a specific user's needs.
- FAILED: The main functionalities are divided by pages, but users might want to have the chatbot on the side. This option is currently not supported.

3. **Discuss if you changed the schema or source of the data for your application.**

- Our initial proposal listed two datasets: one from Kaggle that contained tuition, diversity, and post-grad salaries for US universities, and one from the US Department of Education, which contained information on US universities organized by field of study. From these, we ended up only using the Kaggle dataset in our final implementation
- We included three new datasets: data.mendeley.com with RateMyProfessor information for each university, fivethirtyeight on Github containing salary and demographic

information of college majors, and worldpopulationreview.com which listed the crime rate, population, and happiness level of each state in the US.

4. **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

- The ER diagram and table implementations submitted on Canvas did not change much in our final design, except for renaming some fields. Although, they went through significant changes before being submitted. Our first draft had “School” with just a couple of fields (like the name and state). And then 3 extra tables called “Enrollment”, “CareerPay”, and “TuitionCost”. However, we decided that all of these attributes are relevant to the School table and it would not make sense to split them up into separate tables. So, our final design merged School, Enrollment, CareerPay, and TuitionCost into one School table with many attributes.
- We also added RateMyProfessor ratings, Diversity, and State tables in direct relation to School. We think this is very suitable for our application because all of these seem to be very relevant factors for recommending a certain school to the user. The type of relationships (one-to-many, etc.) seem appropriate for each category as explained in our ER diagram.
- We also added a Major table in relation to RateMyProfessor ratings in order to gather more information about the various fields that professors teach in. For example, the user may want to know whether there are good professors who teach Aerospace Engineering because it is high-paying and high in demand. Therefore, it would be useful to link good professors with information about their respective majors.
- Lastly, we added a Companies table in relation to State. Certain states have a lot of large companies with high salaries, so a user may be interested in finding that information specific to the state.

5. **Discuss what functionalities you added or removed. Why?**

- We added an implementation of ChatGPT so that a user can ask questions about colleges and anything related to school. We thought this would elevate the use of our application and make it different from other college search engines as it is a one stop site for both open ended questions and analytics on colleges.
- We removed the functionality where the user can check checkboxes to filter through the college list better, because it is very difficult to do so. So in response we implemented the Chat functionality to make sure any questions users have will be answered.

6. **Explain how you think your advanced database programs complement your application.**

- Advanced database programs allow us to filter data based on more sophisticated criteria, which enables us to calculate scores based on several weighted factors instead of one category at a time. For instance, our procedure calculates a score that combines Rate My Professor ratings (weighted by x2) with Crime Rate (weighted by x1). This feature gives users the option to rank colleges based on multiple criteria simultaneously. Additionally, we implemented a trigger that automatically raises/lowers the happiness score if the crime rate in the relevant state is updated, making the data dynamic.

7. **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**

- **Khatija:** A technical challenge we faced initially was ensuring that everyone had access to the Google Cloud Platform so each team member could access and query the data on their local machines. After some research and asking others, we learned that each device is linked to an IP address so we manually added everyone's IP addresses to the GCP. Additionally, we learned how to connect the MYSQL Workbench by creating an account and linking the IP address of the host server. Overcoming this initial challenge helped us work on more difficult queries.
- **Brad:** Using Material UI is extremely difficult, because it requires a lot of hand-adjusting of parameters and what column exists and which doesn't. Here is the solution that we came up with (extrapolate further to add more Table Cells based on what the application requires). This integrates daily well with node.js, and even though adding each column is still a pain, I think it is very human readable and clear what each function is doing.

```
const renderTable = () => (
 <TableContainer component={Paper}>
   <Table>
     <TableHead>
       <TableRow>
         {(operation === 'getAll' || operation === 'get' || operation === 'post' ||
operation === 'put' || operation === 'delete' || operation === 'lowCrime' || operation
== 'highRateMyProf' || operation == 'lowCrimeHighProf') && (
           <TableCell>School Name</TableCell>
         )}
```

- **Tiffany:** While testing the SQL trigger for our advanced database program using MySQL Workbench, we initially tried to write our code in the same place we would write SQL queries, but we ended up getting error messages. After searching up tutorials, we realized that we needed to write the trigger code in a separate designated location, and were able to successfully run the code by following along with a step-by-step video. When trying to navigate through new and unfamiliar interfaces like MySQL, it can be really helpful and efficient to find guided tutorials to help you work through any specific task you are trying to accomplish.

- **Esther:** Data Cleaning is an absolutely critical part of the project. I remember uploading the data files and then querying for information about Colorado by asking `WHERE Code=`CO`. It showed no results and I thought my query was wrong for the longest time. But after double checking the data, I found that the columns had data with a space at the front. So the query was unable to find results because of an additional space. I think we tend to get caught up in writing code, so make sure to keep your mind open to check for other possible bugs.

8. **Are there other things that changed comparing the final application with the original proposal?**

- We have decided to not include a "search" engine. We have tried to implement it, but it is very clunky, and without advanced search algorithms we can't really hope to make it actually useful. Therefore we decided against it.

9. **Describe future work that you think, other than the interface, that the application can improve on**

- In the future, we'd like to fully implement a filter checkbox search for every cross variation of the criteria. We implemented an algorithmic check against crime rates and RateMyProfessor ratings, but we would want to do that for all of the different criteria. Additionally, if we had more time, we'd like to make the UI prettier and more organized so that it matched our original vision of the application.

10. **Describe the final division of labor and how well you managed teamwork.**

- Khatija and Tiffany worked on finding and cleaning the datasets that we wanted to implement into the database. With these datasets, Esther was able to implement the database. Esther and Brad both worked on the simple and complex queries. Brad created a React application with a frontend and backend connected to the GCP. Khatija, Tiffany, and Esther worked on writing the CRUD commands, trigger, and procedure together. Brad then was able to implement the SQL code onto the application and created functionality (using buttons) so we could see the progress on the frontend. Altogether, we managed our teamwork mainly by focusing on dividing the work based on everyone's strengths and interests. If any one of us came across an issue or was unable to do something, we'd find time to work together and conquer the issue together. We made active efforts to reach out to each other by setting meetings between stages to complete the tasks. We additionally would stay after class to ask for additional help and attend the class workshops intended to help us out with the project.

Rubric: This stage is worth 10%. You are graded by completeness and correctness. The rubric is as follows:

1. Each missing section of the report (-1%), if there are no changes or nothing to discuss, state it.
1. The report is not well-written or does not provide enough details (-4%)
2. Missing video (-2%)
1. Too short, too long video (-1%)
2. The video does not provide a comprehensive overview of the application (-2%)
3. **The report should be placed in the docs folder, and a readme file containing the link to the video.** (-1% if missing either component in the folder)
4. **You should tag your release and submit it on canvas. Failure to tag your release will result in a 1% deduction.**
5. Failure to submit your catme survey will result in a 1% deduction from the stage.